Gradient-Guided RC Weighting for Timing-Driven Global Routing

Liang Xiao[1], Qinkai Duan[1], Leilei Jin[1], Jinwei Liu[2], Tsung-Yi Ho[1],
Evangeline F.Y. Young[1], and Martin D. F. Wong[2]
[1]The Chinese University of Hong Kong
[2]Hong Kong Baptist University

March 16, 2026

Content

- Introduction
  - Global routing
  - Timing model
- Algorithm
- Experiments

1 Introduction

Global routing:

- In global routing, the router connects nets in a coarsened grid graph.
- We use the augmented pattern routing algorithm introduced in CUGR2[1], and use the code base of InstantGR[2].

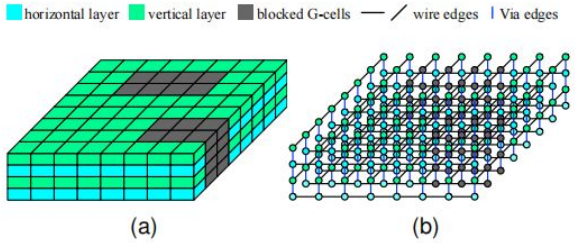


Fig. 1: Global routing grid graph. (a) G-cell partitioning. (b) Grid graph.

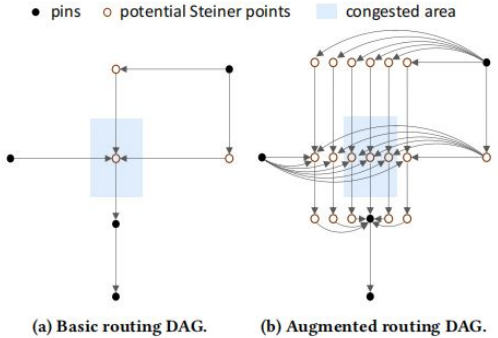
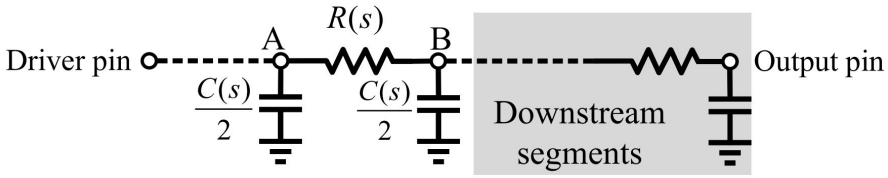


Figure 1: Global routing using routing DAG.

[1] Jinwei Liu and Evangeline F.Y. Young. 2023. EDGE: Efficient DAG-based GlobalRouting Engine. In *ACM/IEEE Design Automation Conference (DAC)*. 1–6.
[2] Liang Xiao, Shiju Lin, Jinwei Liu, Qinkai Duan, Tsung-Yi Ho, and Evangeline F. Y.Young. 2025. InstantGR: Scalable GPU Parallelization for 3-D Global Routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*(2025), 1–1. doi:10.1109/TCAD.2025.3573685

Timing model:



- We use Elmore model to model the net delay.

$$Delay(s) = R(s) \cdot (C(s)/2 + C_{\text{down}}(s))$$

- We focus on the net delay, and use a simplified timing model that calculates the arrival time of a cell's output pin as the maximum arrival time of all inputs with a cell delay.

$$Arr(p_{out}) = \max_{p_i \in In}(Arr(i)) + D_c$$

Content

- Introduction
- Algorithm
  - Overview
  - Arc-level Gradient
  - Segment-level Gradient
  - Timing-aware DAG-routing
  - Power optimization
- Experiments

1 Algorithm

Overview:

Key idea: Prioritize the most "important" wire segments.

To achieve this, we want to know the influence of a segment's R,C on the timing results.

We then consider in two levels:

- **Arc-level Gradient:** To estimate how timing Arcs' delay influence endpoints' slack.
- **Segment-level Gradient:** To estimate how segments' R,C influence Arcs' delay.

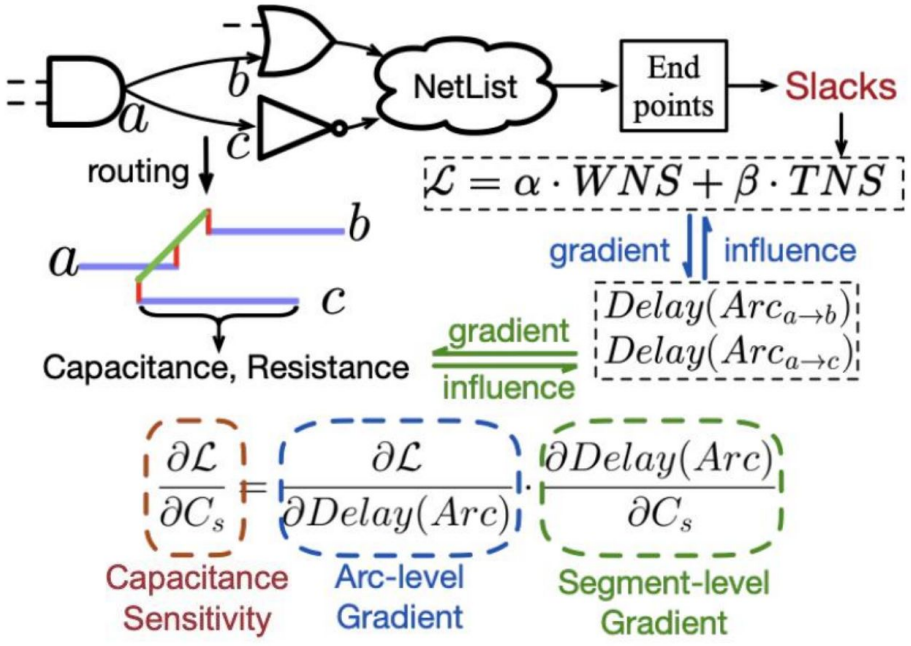


Figure 2: Example of the partial gradient of timing objective with respect to the capacitance of segment s.

1 Algorithm

Arc-level Gradient:

Before calculating Arc-level gradient, we estimate the influence of endpoints arrival time

For TNS:

Each negative slack endpoint contribute once to the TNS. (negative slack endpoints are more important):

$$\frac{\partial TNS}{\partial Arr(e)} = \begin{cases} 1, & \text{if } Arr(e) > Req(e), \\ 0, & \text{else.} \end{cases}$$

For WNS:

Inspired by [3] , we use LSE function to approximate WNS:

$$\begin{aligned} WNS &= \min_{e \in END}(Slack(e)) \\ &= -\max_{e \in END}(-Slack(e)) \approx -LSE(-Slack(END)) \\ &= -(M + log(\sum_{e \in END} \tau \cdot exp(\frac{-Slack(e) - M}{\tau}))) \end{aligned}$$

According to this approximation, the gradient respect to an endpoint's arrival time is as follows: (Endpoints with more negative slack are more important for the WNS):

$$\begin{aligned} \frac{\partial WNS}{\partial Arr(e)} &= -\frac{\partial WNS}{\partial Slack(e)} \approx -\frac{\partial - LSE(-Slack(END))}{\partial Slack(e)} \\ &= \frac{exp(\frac{-Slack(e_i) - M}{\tau})}{\sum_{e_i \in END} exp(\frac{-Slack(e_i) - M}{\tau})} \end{aligned}$$

[3] Yi-Chen Lu, Zhizheng Guo, Kishor Kunal, Rongjian Liang, and Haoxing Ren.2025. INSTA: An Ultra-Fast, Differentiable, Statistical Static Timing Analysis Engine for Industrial Physical Design Applications. In 2025 62th *ACM/IEEE Design Automation Conference (DAC).*

Arc-level Gradient:

We then backpropagate the “importance” from endpoints to the start points.

For a cell with multiple input pins, the pins with more negative slack are more important for the arrival time of the output pin:

$$Arr(p_{out}) = \max_{p_i \in In}(Arr(i)) + D_c$$

$$\frac{\partial Arr(p_{out})}{\partial Arr(p_i)} \approx \frac{\partial LSE(In)}{\partial Arr(p_i)} = \frac{exp(\frac{Arr(p_i)-M}{\tau})}{\sum_{p_j \in In} exp(\frac{Arr(p_j)-M}{\tau})} \approx \frac{exp(\frac{-Slack(i)-M'}{\tau})}{\sum_{j \in In} exp(\frac{-Slack(j)-M'}{\tau})}$$

For a net, the arrival time of the driver pin influences all sink pins. the gradient with respect to the driver pin’s arrival time is the sum of the gradient at all sink pins:

$$\frac{\partial \mathcal{L}}{\partial Arr(p_0)} = \sum_{p_{out} \in Sinks(n)} \frac{\partial \mathcal{L}}{\partial Arr(p_{out})}$$

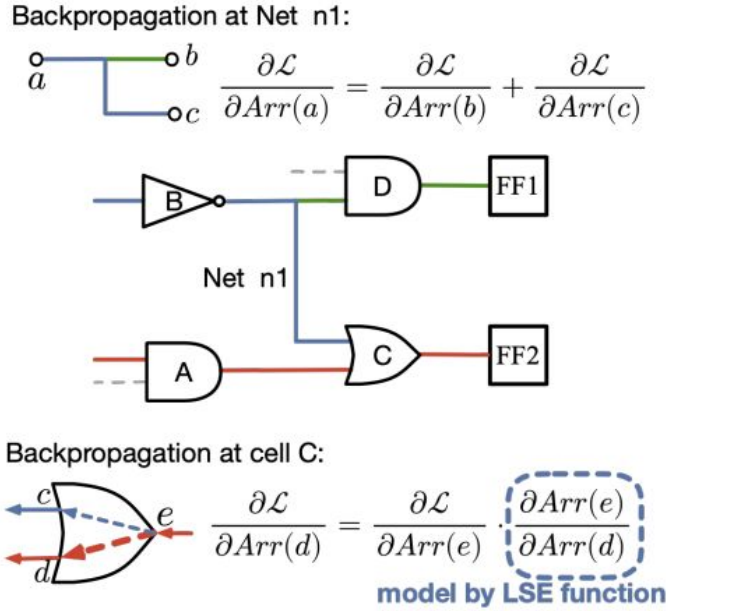


Figure 3: The process of backpropagation. The red, blue, and green paths are negative slack paths with the first, second, and third worst slack values, respectively.

1 Algorithm

Segment-level Gradient:

According to Elmore delay model. The delay of a net arc $Arc_{p_0 \to p_{out}}$ is the sum of all segments delay on the path:

$$Delay(Arc_{p_0 \to p_{out}}) = \sum_{s \in P(p_0 \to p_{out})} Delay(s).$$

$$Delay(s) = R(s)\left(C(s)/2 + C_{\text{down}}(s)\right)$$

Differentiating these two formula and grouping terms yields:

$$\frac{\partial Delay(Arc_{p_0 \to p_{out}})}{\partial C(s_i)} = \frac{R(s_i)}{2} \mathbf{1}\{s_i \in U_{p_{out}}\} + \sum_{s \in U_{p_{out}} \cap U_{s_i}} R(s).$$

$$\frac{\partial Delay(Arc_{p_0 \to p_{out}})}{\partial R(s_i)} = \left(C(s_i)/2 + C_{\text{down}}(s_i)\right) \mathbf{1}\{s_i \in U_{p_{out}}\}.$$

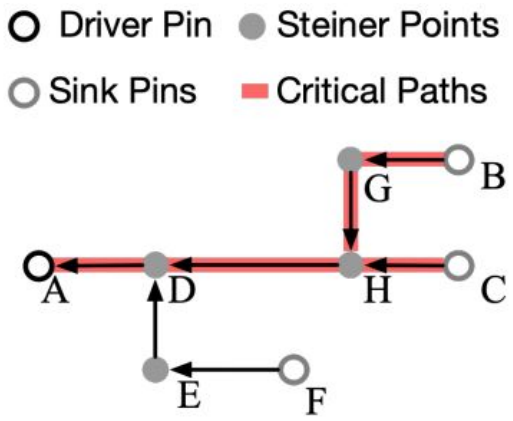


Figure 4: The routing graph of a 4-pin net.

R,C are estimated by multiplying wirelength and unit length capacitance, unit length resistance.
The tree structure is based on Flute.

Segment-level Gradient:

Gradient on multi-sink nets:

$$\frac{\partial \mathcal{L}}{\partial C(s_i)} = \sum_{p_{out} \in \mathcal{S}} w_{p_{out}} \frac{\partial Delay(Arc_{p_0 \to p_{out}})}{\partial C(s_i)} \tag{22}$$

$$= \sum_{p_{out} \in \mathcal{S}} w_{p_{out}} \left(\frac{R(s_i)}{2} \mathbf{1}\{s_i \in U_{p_{out}}\} + \sum_{s \in U_{p_{out}} \cap U_{s_i}} R(s) \right) \tag{23}$$

$$= \left(\sum_{p_{out} \in \mathcal{S}(s_i)} w_{p_{out}} \right) \cdot \frac{R(s_i)}{2} + \sum_{s \in U_{s_i}} \left(R(s) \sum_{p_{out} \in \mathcal{S}(s)} w_{p_{out}} \right) \tag{24}$$

$$\frac{\partial \mathcal{L}}{\partial R(s_i)} = \sum_{p_{out} \in \mathcal{S}} w_{p_{out}} \frac{\partial Delay(Arc_{p_0 \to p_{out}})}{\partial R(s_i)} \tag{25}$$

$$= \left(\sum_{p_{out} \in \mathcal{S}(s_i)} w_{p_{out}} \right) \cdot \left(\frac{C(s_i)}{2} + C_{down}(s_i) \right). \tag{26}$$

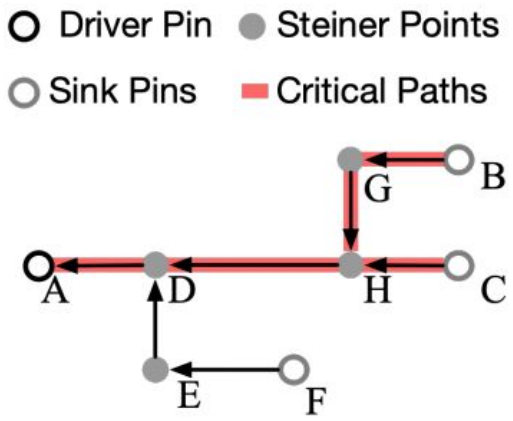


Figure 4: The routing graph of a 4-pin net.

Timing-aware DAG-routing:

With Arc-level gradient and Segment-level gradient, we can calculate a “Resistance sensitivity” and a “Capacitance sensitivity” for each wire segment.

We integrate the timing optimization into global routing by modifying the segment cost:

$$Cost(s) = Overflow(s) + \gamma \cdot \left(\frac{\partial \mathcal{L}}{\partial C(s)} \cdot C(s) + \frac{\partial \mathcal{L}}{\partial R(s)} \cdot R(s)\right)$$

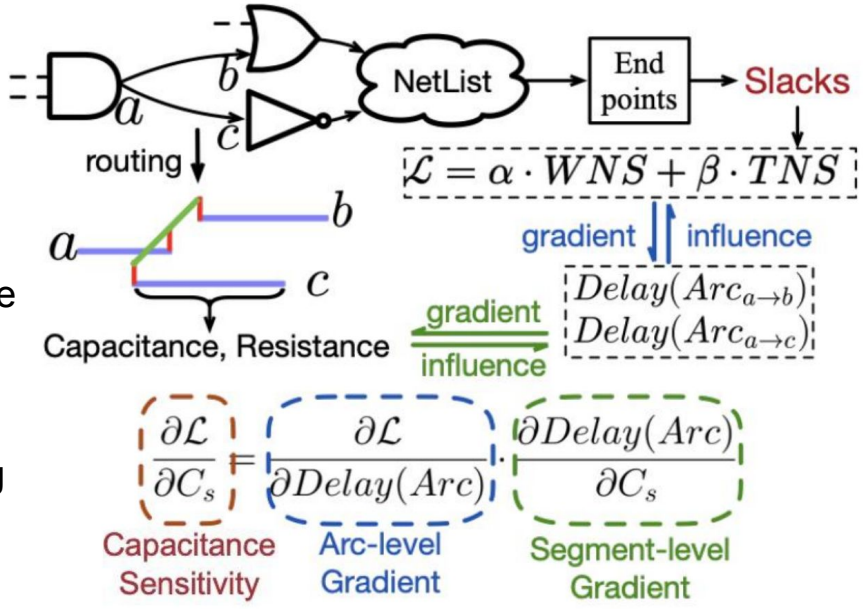


Figure 2: Example of the partial gradient of timing objective with respect to the capacitance of segment s.

1 Algorithm

Power optimization:

We optimize the power via minimizing the switching power, which can be calculated as follows:

$$P_{\text{switching}} = \alpha \cdot C_{\text{load}} \cdot V^2 \cdot f$$

α is a switching activity factor representing the probability of signal transitions per clock cycle.
C_{load} is the load capacitance.
V is the supply voltage, f is the frequency of the clock.

We use OpenROAD[4] to calculate the following power weight and add it to the original capacitance sensitivity.

$$w_p = \alpha \cdot V^2 \cdot f$$

[4] 2024. OpenROAD. https://github.com/The-OpenROAD-Project/OpenROAD.git. Accessed: April 2025.

Content

- Introduction
- Algorithm
- Experiments
  - Timing results
  - Power results

1 Algorithm

Timing results:

Compared with the first-place team, we achieve:

- 18.5% TNS improvement
- 14.3% WNS improvement

with a 3.6% congestion overhead

Table 2: Experimental result of total score and runtime.

Benchmark	1st		2nd		3rd		Ours	
	Score	Runtime	Score	Runtime	Score	Runtime	Score	Runtime
0	0.897	6.51	0.640	7.04	0.095	15.88	-0.945	7.24
1	0.562	32.94	0.722	22.07	0.094	37.25	-0.409	24.55
2	-1.995	9.20	-2.077	22.37	-1.824	18.92	-2.144	12.36
3	-0.405	7.49	0.401	11.03	1.370	15.53	0.443	11.80
4	-2.029	133.53	-1.295	153.52	-3.599	266.88	-2.180	103.88
5	0.510	501.61	0.399	839.77	0.714	915.47	0.497	609.10
6	1.780	7.06	1.646	7.15	1.519	23.74	1.461	11.35
7	0.860	37.23	0.426	27.79	0.700	44.80	0.424	33.70
8	1.402	9.42	4.375	20.63	1.364	19.43	1.391	16.32
9	0.358	7.01	2.088	11.47	1.817	15.88	1.205	11.75
10	1.333	135.70	1.233	161.55	3.117	247.71	1.346	103.78
11	2.327	491.11	2.371	766.29	2.678	1965.67	2.313	550.58
Average	0.467	**114.90**	0.911	170.89	0.670	298.93	**0.284**	124.70
Ratio	1.645	**0.921**	2.209	1.370	1.191	2.397	**1.000**	1.000

Table 3: Experimental results of the top-3 teams of ISPD2025 Contest and our algorithm.

Bench	1st				2nd				3rd				Ours			
	TNS	WNS	Power	Congestion	TNS	WNS	Power	Congestion	TNS	WNS	Power	Congestion	TNS	WNS	Power	Congestion
0	-1325.78	-0.432	0.646	5846461	-1340.19	-0.398	0.646	5902618	-1138.37	-0.381	0.646	8003814	-1026.76	-0.379	0.646	6438399
1	-10366.19	-0.420	3.054	21332547	-10774.87	-0.424	3.053	20468992	-9344.33	-0.407	3.054	25412296	-8666.18	-0.401	3.053	22573155
2	-521344.56	-56.919	2.941	13315606	-514660.28	-55.841	2.940	13686713	-514674.25	-55.914	2.938	15687876	-515533.69	-55.926	2.941	13056103
3	-1610.01	-0.379	0.143	3019425	-2225.46	-0.489	0.145	2478317	-2524.33	-0.520	0.145	3469019	-2085.02	-0.461	0.145	2910471
4	-20473.81	-0.305	7.692	55005746	-21709.13	-0.347	7.734	49071539	-19338.70	-0.275	7.553	97506552	-19042.52	-0.283	7.673	65052484
5	-59712.49	-0.266	23.450	236920427	-62459.98	-0.315	23.578	192012708	-59533.47	-0.276	23.220	289712328	-54359.36	-0.264	23.320	252306571
6	-650.10	-1.756	0.156	4349152	-149.52	-0.737	0.156	4651055	-162.38	-0.633	0.156	4386728	-45.87	-0.431	0.156	4374734
7	-2859.79	-3.984	0.305	22567636	0.00	0.000	0.305	21307154	-2936.03	-2.747	0.305	20564539	0.00	0.000	0.305	21213727
8	0.00	0.000	0.136	14061156	0.00	0.000	0.136	16352692	0.00	0.000	0.136	13637302	0.00	0.000	0.136	13936159
9	-1356.07	-0.342	0.143	2232008	-1920.72	-0.472	0.145	2122543	-1806.79	-0.440	0.145	2168599	-1600.96	-0.392	0.144	2246898
10	-32016.10	-0.433	8.369	52783071	-34846.05	-0.460	8.359	49722314	-26801.44	-0.373	7.942	132027917	-29386.86	-0.421	8.349	55695359
11	-42564.08	-0.222	24.244	242737068	-50372.44	-0.294	24.380	214110807	-39516.23	-0.200	23.968	335020596	-37846.03	-0.216	24.152	261589161
Ratio	1.000	1.000	1.000	1.000	0.954	0.975	1.004	**0.956**	0.969	0.937	**0.995**	1.302	**0.815**	**0.857**	1.000	1.036

Power results:

Enabling the power optimization, our router further improves:

- 10.6% switching power
- 4.1% total power
- 0.884 contest score

with an 8.3% congestion overhead.

Table 4: Experimental results of power optimization.

Bench	w/o power optimization				w/ power optimization			
	P_s	P_t	Congestion	Score	P_s	P_t	Congestion	Score
0	0.0009	0.6461	6438399	-0.945	0.0009	0.6460	6443602	-0.977
1	0.0772	3.0531	22573155	-0.409	0.0773	3.0531	22576420	-0.409
2	0.1621	2.9406	13056103	-2.144	0.1561	2.9345	13175395	-2.285
3	0.0488	0.1447	2910471	0.443	0.0454	0.1417	3069599	-0.349
4	3.2985	7.6727	65052484	-2.180	3.0019	7.3692	72775512	-7.962
5	10.2331	23.3196	252306571	0.497	9.7320	22.8011	269961040	0.426
6	0.0001	0.1561	4374734	1.461	0.0001	0.1561	4376830	1.462
7	0.0024	0.3048	21213727	0.424	0.0024	0.3048	21180832	0.424
8	0.0013	0.1363	13936159	1.391	0.0013	0.1363	13972036	1.394
9	0.0476	0.1443	2246898	1.205	0.0447	0.1410	2416448	1.172
10	3.7161	8.3489	55695359	1.346	3.2413	7.8588	64429910	0.235
11	10.9162	24.1525	261589161	2.313	9.4767	22.6697	292718663	-0.332
Average	2.3754	5.9183	**60116102**	0.284	**2.1483**	**5.6843**	65591357	**-0.600**
Ratio	1.106	1.041	**0.917**		**1.000**	**1.000**	1.000	

Thank you